CLAIMS

1. In a wireless communications device, a method for reorganizing software instructions stored in a memory, the method comprising:

storing wireless device system software in a plurality of current code sections;

receiving a new code section;

resizing current code sections;

arranging the new code section with the current code sections to form updated system software for the wireless device; and,

executing the updated system software.

2. The method of claim 1 further comprising:

identifying a current code section for updating; and,

wherein arranging the new code section with the current code sections to form updated system software includes replacing the identified current code section with the new code section.

3. The method of claim 2 further comprising:

forming the system software into a first plurality of symbol libraries, each symbol library comprising at least one symbol; and,

arranging the first plurality of symbol libraries into a second plurality of code sections.

4. The method of claim 3 wherein receiving a new code section includes receiving the new code section via a wireless communications device air interface.

5. The method of claim 4 wherein arranging the first plurality of symbol libraries into a second plurality of code sections includes starting symbol libraries at the start of code sections;

wherein storing wireless device system software in a plurality of current code sections includes storing the start of code sections at corresponding start addresses; and,

the method further comprising:

maintaining a code section address table cross-referencing code section identifiers with corresponding start addresses.

6. The method of claim 5 wherein arranging the first plurality of symbol libraries into a second plurality of code sections includes arranging symbols to be offset from their respective code section start addresses; and

the method further comprising:

maintaining a symbol offset address table cross-referencing symbol identifiers with corresponding offset addresses, and corresponding code section identifiers.

7. The method of claim 6 wherein storing the start of code sections at corresponding start addresses includes:

creating a second plurality of contiguously addressed memory blocks;

identifying each memory block with a corresponding code section; and,

storing code sections in the identified memory blocks.

8. The method of claim 7 wherein arranging the first plurality of symbol libraries into a second plurality of code sections includes sizing the code sections to accommodate arranged symbol libraries; and,

wherein creating a second plurality of contiguously addressed memory blocks includes sizing memory blocks to accommodate corresponding code sections.

9. The method of claim 8 wherein sizing memory blocks to accommodate corresponding code sections includes sizing the code sections to accommodate sizes larger than the arranged symbol libraries.

10. The method of claim 9 wherein resizing current code sections includes resizing the memory blocks in which corresponding resized code sections are stored.

11. The method of claim 10 wherein receiving a new code section includes receiving a new code section having a first size;

wherein identifying a current code section for updating includes identifying a current code section having a second size, less than the first size; and,

wherein resizing the memory blocks in which corresponding resized code sections are stored includes:

increasing the size of a memory block associated with the identified current code section to at least the first size; and,

replacing the identified current code section, stored in the corresponding memory block, with the new code section.

12. The method of claim 11 wherein resizing the memory blocks in which corresponding resized code sections are stored includes, in response to measuring the size of symbol libraries arranged within the corresponding code sections, resizing code sections to more closely match the symbol library sizes arranged within.

13. The method of claim 12 wherein resizing code sections to more closely match the symbol library sizes arranged within includes optimally resizing code sections to further subsequent code section resizing and updating operations.

14. The method of claim 12 further comprising:

using the start addresses from code section address table, measuring the current code sections sizes;

using the symbol offset addresses from the symbol offset address table, measuring the size of the symbol libraries arranged within corresponding code sections.

15. The method of claim 14 wherein forming the system software into a first plurality of symbol libraries includes forming end symbols to signify the end of symbol libraries; and,

wherein measuring the size of the symbol libraries arranged within corresponding code sections includes using the offset addresses of the end symbols to measure the size of symbol libraries.

16. The method of claim 14 wherein forming the system software into a first plurality of symbol libraries includes forming size symbols to signify the size of symbol libraries; and,

wherein measuring the size of the symbol libraries arranged within corresponding code sections includes accessing the size symbols to measure the size of symbol libraries.

17. The method of claim 15 further comprising:

following the resizing of the code sections, changing the code section start addresses.

18. The method of claim 16 further comprising:

measuring the size of the new code section; and,

in response to measuring the size of symbol libraries arranged within corresponding code sections, and measuring the size of the new code section, determining if the new code section can be arranged with the current code sections.

19. The method of claim 18 wherein determining if the new code section can be arranged with the current code section includes determining the size of unused memory blocks; and,

wherein arranging the new code section with the current code sections to form updated system software for the wireless device includes storing the new code section in the unused memory block, if the size of the unused memory block is greater than, or equal to the new code section size.

20. The method of claim 18 further comprising:

calculating the code section sizes;

in response to calculating the code section sizes, generating a compaction schedule;

temporarily moving code sections into a file system section; and,

wherein arranging the new code section with the current code sections to form updated system software for the wireless device includes storing the code sections from the file system section into the memory blocks to maintain contiguous addressing, in response to the compaction schedule.

21. The method of claim 20 further comprising:

in response to changing the start addresses of code sections stored within the memory blocks, generating an updated code section address table; and,

wherein executing the updated system software includes using the updated code section address table after arranging the new code section with the current code sections.

22. The method of claim 21 wherein receiving a new code section includes receiving a new code section with an updated symbol offset address table; and,

wherein executing the updated system software includes using the updated symbol offset address table after arranging the new code section with the current code sections.

23. The method of claim 11 wherein receiving a new code section includes receiving a compaction instruction set including code section resizing instructions and a compaction schedule; and,

wherein resizing current code sections includes resizing code sections in response to the code section resizing instructions.

24. The method of claim 23 further comprising:

temporarily moving code sections into a file system section; and,

wherein arranging the new code section with the current code sections to form updated system software for the wireless device

includes storing the code sections from the file system section into memory blocks to maintain contiguous addressing, in response to the compaction schedule.

25. The method of claim 24 wherein receiving a new code section includes receiving a new code section with an updated code section address table and an updated symbol offset address table; and,

wherein executing the updated system software includes using the updated code section address table and updated symbol offset address table after arranging the new code section with the current code sections.

26. The method of claim 6 further comprising:

loading the code section address table and symbol offset address table into a volatile memory;

in response to loading the code section address table and symbol offset address table into the volatile memory, executing system software;

resetting the wireless communications device;

in response to resetting, loading the updated code section address table and the updated symbol offset address table into volatile memory; and,

wherein executing the updated system software includes executing the updated system software in response to loading the

updated code section address table and updated symbol offset address table into memory.

27. The system of claim 26 wherein resizing current code sections includes suspending the operation of the system software.

28. In a wireless communications device, a method for reorganizing software instructions stored in a memory, the method comprising:

storing wireless device system software in a plurality of current code sections with the start of code sections at corresponding start addresses by creating a second plurality of contiguously addressed memory blocks, identifying each memory block with a corresponding code section, and storing code sections in identified memory blocks;

receiving a new code section via a wireless communications device air interface;

identifying a current code section for updating;

calculating the code section sizes;

in response to calculating the code section sizes, generating a compaction schedule;

resizing current code sections;

following the resizing of the current code sections, changing the code section start addresses;

temporarily moving code sections into a file system section;

replacing the identified current code section with the new code section by storing the code sections from the file system section into memory blocks to maintain contiguous addressing, in response to the compaction schedule; and,

executing the updated system software.

29. In a wireless communications device, a system for reorganizing software instructions stored in a memory, the system comprising:

a code storage section memory including executable wireless device system software differentiated into a plurality of current code sections;

a file system section memory for receiving new code sections;

a compactor to resize current code sections; and

wherein the arrangement of new code sections with the current code sections in the code storage section forms updated system software.

30. The system of claim 29 wherein the file system section receives a compaction instruction set with instructions for identifying a current code section for updating; and,

wherein the compactor replaces the identified current code section in the code storage section with the new code section.

31. The system of claim 30 wherein the code storage section comprises a first plurality of symbol libraries, each symbol library comprising at least one symbol, with the first plurality of symbol libraries being arranged into a second plurality of code sections.

32. The system of claim 31 further comprising:

an airlink interface to receive new code sections; and,

wherein the file system section receives the new code section via a wireless communications device airlink interface.

33. The system of claim 32 wherein the code storage section symbol libraries start at the start of code sections; and,

wherein the code storage section includes a first table code section with a code section address table for cross-referencing code section identifiers with corresponding start addresses.

34. The system of claim 33 wherein the code storage section includes symbols arranged to be offset from their respective code section start addresses; and,

wherein the code storage section includes a second table code section with a symbol offset address table for cross-referencing symbol identifiers with corresponding offset addresses, and corresponding code section identifiers.

35. The system of claim 34 wherein the code storage section includes a second plurality of contiguously addressed memory blocks identified with the corresponding second plurality of code sections.

36. The system of claim 35 wherein the code storage section includes code sections sized to accommodate the symbol libraries arranged within, and memory blocks sized to accommodate the corresponding code sections.

37. The system of claim 36 wherein the code storage section includes code sections sized to accommodate sizes larger than the symbol libraries arranged within.

38. The system of claim 37 wherein the compactor resizes the memory blocks in which corresponding resized code sections are stored in the code storage section.

39. The system of claim 38 wherein the file system section receives a new code section having a first size;

wherein compaction instruction set identifies a current code section having a second size, less than the first size; and,

wherein the compactor increases the size of a memory block associated with the identified current code section to at least the first size, and replaces the identified current code section, stored in the corresponding memory block, with the new code section.

40. The system of claim 39 wherein the compactor determines the size of symbol libraries arranged within the corresponding code sections, and resizes code sections to more closely match the symbol library sizes arranged within.

41. The system of claim 40 wherein the compactor optimally resizes code sections to further subsequent code section resizing and updating operations.

42. The system of claim 40 wherein the compactor accesses start addresses from code section address table, to measure the code sections sizes; and,

wherein the compactor accesses symbol offset addresses from the symbol offset address table, to measure the size of the symbol libraries arranged within corresponding code sections.

43. The system of claim 42 wherein the code storage section includes symbol libraries with end symbols to signify the end of symbol libraries; and,

wherein the compactor uses the end symbol offset addresses to measure the size of symbol libraries arranged within corresponding code sections.

44. The system of claim 42 wherein the code storage section includes symbol libraries with size symbols to signify the size of symbol libraries; and,

wherein the compactor accesses the size symbols to measure the size of symbol libraries arranged within corresponding code sections.

45. The system of claim 43 wherein the compactor changes the start addresses of code sections stored in the code storage section, after resizing the code sections.

46. The system of claim 45 wherein the compactor measures the size of the new code section in the file system section and determines if the new code section can be arranged with the current code sections in the code storage section, in response to measuring the size of symbol libraries arranged within corresponding code sections and measuring the size of the new code section.

47. The system of claim 45 wherein the file system section receives an compaction instruction set including the size of the new code section; and,

wherein the compactor accesses the compaction instruction set to determine the size of the new code section in the file system section and determines if the new code section can be arranged with the current code sections in the code storage section, in response to measuring the size of symbol libraries arranged within

corresponding code sections and determining the size of the new code section.

48. The system of claim 45 wherein the compactor determines the size of unused memory blocks in the code storage section and stores the new code section in the unused memory block, if the size of the unused memory block is greater than, or equal to the new code section size.

49. The system of claim 45 wherein the compactor calculates the code section sizes and, in response to calculating the code section sizes, generates a compaction schedule;

wherein the file system section temporarily stores code sections from the code storage section; and,

wherein the compactor stores the code sections from the file system section into the code storage section memory blocks to maintain contiguous addressing, in response to the compaction schedule.

50. The system of claim 49 wherein the compactor generates an updated code section address table, in response to changing the start addresses of code sections stored within the memory blocks, and,

wherein the updated system software accesses the updated code section address table after the new code section is arranged with the current code sections.

51. The system of claim 50 wherein the file system section receives a new code section with an updated symbol offset address table; and,

wherein the updated system software accesses the updated symbol offset address table after the new code section is arranged with the current code sections.

52. The system of claim 39 wherein the file system section receives a compaction instruction set including code section resizing instructions and a compaction schedule; and,

wherein the compactor resizes code sections in response to the code section resizing instructions.

53. The system of claim 52 wherein the file system section temporarily stores code sections from the code storage section; and,

wherein the compactor stores the code sections from the file system section into the code storage section memory blocks to maintain contiguous addressing, in response to the compaction schedule.

54. The system of claim 53 wherein the file system section receives a new code section with an updated code section address table and an updated symbol offset address table; and,

wherein the updated system software accesses the updated symbol offset address table and updated code section address table after the new code section is arranged with the current code sections.

55. The system of claim 29 wherein the code storage section and file system section memories are nonvolatile memories.

56. The system of claim 34 further comprising:

a volatile memory including the first and second table code sections loaded from the code storage section for executing the system software; and,

wherein the arrangement of new code sections with the current code sections in the code storage section forms updated system software, following a reset of the wireless communications device to load the updated code section address table and updated symbol offset address table in the volatile memory.

57. The system of claim 56 wherein the compactor suspends the execution of the system software.

58. In a wireless communications device, a system for reorganizing software instructions stored in a memory, the system comprising:

a code storage section memory including executable wireless device system software differentiated into a plurality of

current code sections with the start addresses identified with a plurality of contiguously addressed memory blocks;

a file system section for receiving new code sections, via an airlink interface, including a compaction instruction set identifying the current code section for updating;

a compactor to calculate the code section sizes, generate a compaction schedule, resize current code sections, temporarily move code sections into a file system section and replace the identified current code section with the new code section by storing the code sections into memory blocks to maintain contiguous addressing; and,

wherein the arrangement of new code sections with the current code sections in the code storage section forms updated system software.